ERDC MSRC PET Technical Report No. 01-26

Comparison of Multiblock Grid and Domain Decomposition in Coastal Ocean Circulation Modeling

by

Phu Luong
Clay P. Breshears
Le N. Ly

15 May 2001

Work funded by the Department of Defense
High Performance Computing Modernization Program
U.S. Army Engineer Research and Development Center
Major Shared Resource Center through

Programming Environment and Training

Supported by Contract Number: DAHC94-96-C0002
Computer Sciences Corporation

Comparison of Multiblock Grid and Domain Decomposition in Coastal Ocean Circulation Modeling

Phu Luong
Engineer Research and Development Center
Major Shared Resource Center
Vicksburg, MS 39180
phu@nrcmail.wes.hpc.mil
fax: 601-634-3808

and

Clay P. Breshears
KAI Software
A Division of Intel Americas, Inc.
clay.breshears@intel.com
fax: 217-356-5199

and

Le N. Ly
Department of Oceanography
Naval Postgraduate School
Monterey, CA 93943
lely@hydro.oc.nps.navy.mil
fax:831-656-2712

Abstract

In coastal ocean modeling, the traditional one-block rectangular grid has large memory requirements and long processing times. For a large ocean domain with complicated coastlines, the number of grid points used in the calculation is often the same or even smaller than the number of unused grid points. These problems have long been a big concern in coastal ocean modeling.

To eliminate these problems, many techniques have been introduced over the years. Among these techniques, domain decomposition, nesting, and multiblock grids are the most frequently used. The Multiblock Grid Princeton Ocean Model (MGPOM) uses the Message Passing Interface (MPI) to parallelize computations by assigning each grid block used to a unique processor. MGPOM is not dependent upon the method of grid-generation and is an ideal tool to measure performance differences between gridding methodologies.

The main focus of this study is a comparison of the grid-generation technique and domain decomposition in terms of load balance and overall performance. The data set used for this comparison will be a 90-day simulation for the U.S. west coast under two 29-block grids, one grid generated by domain decomposition and the other by the multiblock technique.

Keywords: Multiblock Grid Princeton Ocean Model, U.S. west coast simulation, coastal ocean circulation model.

1 Introduction

Over the years, the traditional one-block rectangular grid has been used for ocean circulation modeling. This technology encounters difficulty on computational grids with high resolution because of the large memory and computing requirements. For a large body of water, such as an ocean with complicated coastlines, the number of grid points used in the calculation (water points) is often the same or even smaller than the number of unused grid points (land points).

Domain decomposition can be used to partition the traditional one-block grid into subdomains that reduce the unused grid points, and Message Passing Interface (MPI) [1] can be used to parallelize this type of of computation [2]. Domain decomposition often requires a preprocessing step to determine the most efficient work distribution for the subdomains in order to avoid severe load imbalances. Moreover, in this technique, each domain is allowed to communicate with only one adjacent neighbor subdomain at the interface between the two. This limitation affects the ability to eliminate the land-grid points along a complicated ocean coastline. Along complicated coastlines, subdomains may be composed mostly of land-grid points that can cause load balance inequities, without special treatment in the preprocessing step. One-block grid data arrays, often considered as global arrays, are decomposed into the local array structures for each

subdomain. Bookkeeping of global and local array indices is required for communication between MPI processes. This preprocessing step is sometimes very time-consuming for many ocean circulation models and hydrodynamics models.

Another approach, known as multiblock grid-generation, can be used to reduce the unused grid points and to improve the performance of the model as well. This methodology allows the elimination of blocks composed mainly of land-grid points and the choice of the grid with minimum land-grid points along the coastline. In addition, high horizontal grid resolution in an area of interest can be handled easily. MPI is used for parallelization of the ocean model by assigning each grid block to a unique processor. Workload (number of used grid points) for each block can be determined during the grid-generation process to achieve a more even load balance. Another advantage of the multiblock grid over the domain decomposition is that each MPI process can communicate with more than one adjacent neighbor grid block at the interfaces.

In this study, the MPI parallel code version of the Multiblock Grid Princeton Ocean Model (MGPOM) [3] is used for simulation of the U.S. west coast. Pthreads is also used as second-level parallelism within one routine of the code to improve load imbalance between MPI processes. A brief description of 29-block grids generated by domain decomposition and by the grid-generation technique is presented in

Section 2. Models, as well as data, used for simulation in this study are presented in Section 3. The Pthreads implementaion in the routine PROFQ of the MGPOM code is described in Section 4. Performance results of the parallel codes (MPI-Only and MPI-Pthreads) on the two computational model grids are discussed in Section 5. Conclusions from this study are presented in Section 6.

2 Grids Used

The physical geographic area for this study extends from 116 West to 135 West in longitude and from 30 North to 49 North in latitude (Figure 1). The 4-min resolution grid for this area yields a total number of grid points for a one-block rectangular grid (01BLK) of 81,796 (286×286). The number of used and unused grid points is 56,146 and 25,650, respectively. More than 31 percent of the total grid is unused (Figure 2).

A simple routine is developed for reading in the 01BLK grid and evenly decomposing the grid into equal subdomains. Partitioning the 01BLK grid by six equal amounts along the x-direction and six along the y-direction yields 36 subdomains. Since MGPOM uses four grid line overlaps in each direction for each subdomain, the dimensions for each subdomain are 51×51 (2,601 total grid points). Under this particular partition, 21 subdomains are composed of only water-grid

Figure 1: The U.S. west coast coastline

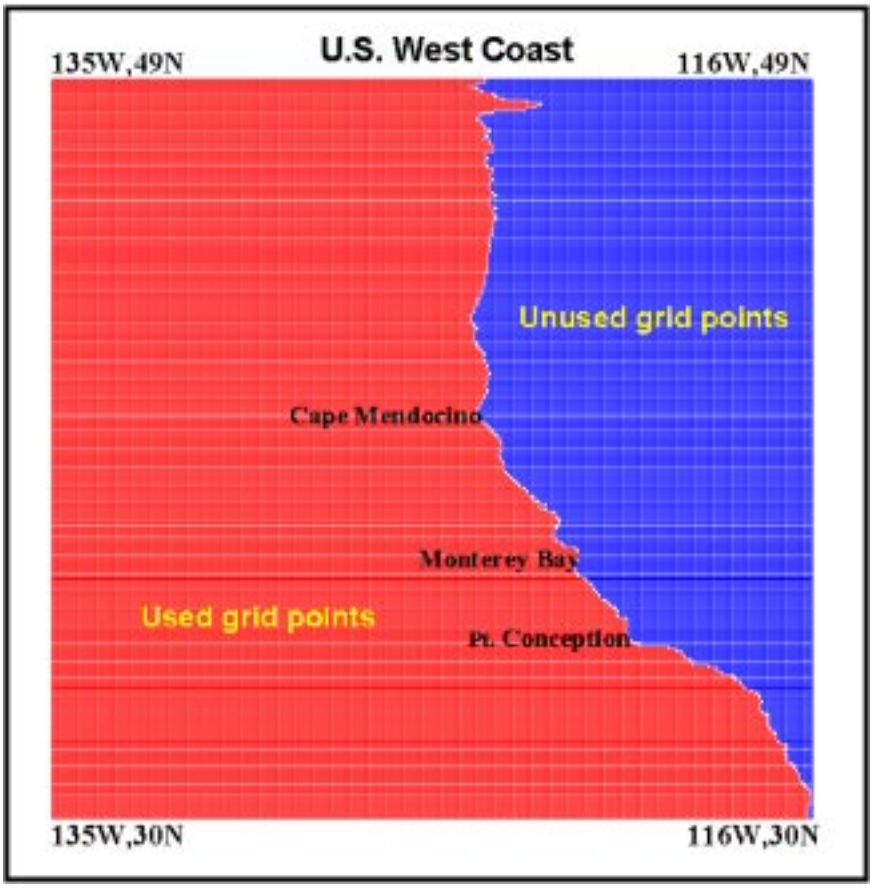

Figure 2: Used and unused grid points

points; 8 subdomains are composed of water-grid and land-grid points; and 7 subdomains are composed of only land-grid points. The 7 subdomain grids with only land-grid points will be neglected in the computations, and the other 29 subdomain grids (referred to as 29DDG) will be used (Figure 3). The total number of unused grid points for this 29DDG grid is 11,015 as compared with 25,650 of the 01BLK grid. The smallest number of used grid points is 129 within this decomposition; this can lead to a severe load imbalance (Figure 4).

As mentioned earlier, the multiblock grid-generation technique has an advantage of selecting grid blocks with minimum land-grid points along the coastline. This is done through the use of the EAGLEView interactive software package [4]. The procedure for generating the 29-block grid (Figure 5) in this technique is given below. This 29-block grid shall be referred to as 29MBG.

First, one uses the Input/Output (I/O) module of EAGLEView to load in the 01BLK grid data and the U.S. west coast coastline data. The unused grid points are then extracted from the 01BLK grid by using the Extract module. The Extract module is then able to extract the remaining part (used grid points) from the domain. This same module is used for decomposing the used grid portion into 18 equal small grid blocks. This yields each grid block with dimensions 47×51 (2,397 total grid points).

Second, the Extract module is used again for generating small grid

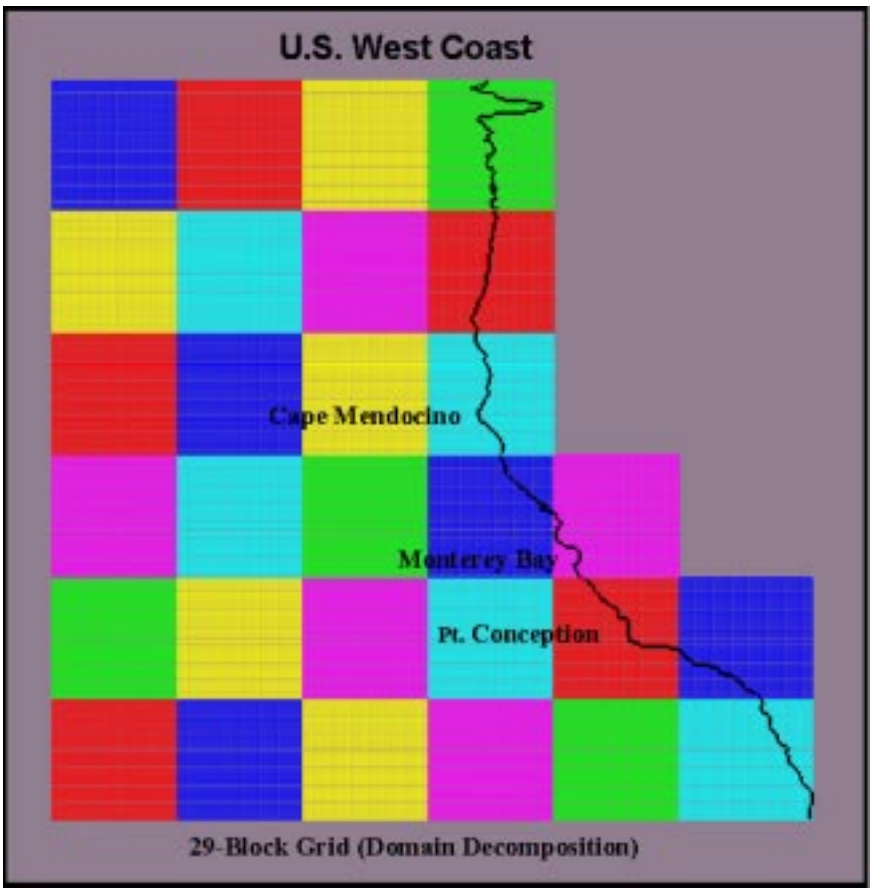

Figure 3: Coastline on the 29DDG grid

blocks along the coastline of the remaining part of the domain. This module allows users to select grid blocks along the coastline such that each grid block contains a minimum number of unused grid points. The List module is then used for checking the dimensions as well as the number of used grid points in each small grid block. Since MGPOM is a multiblock grid code, it allows multiple communication at the interface of each grid block. This property makes it easier to

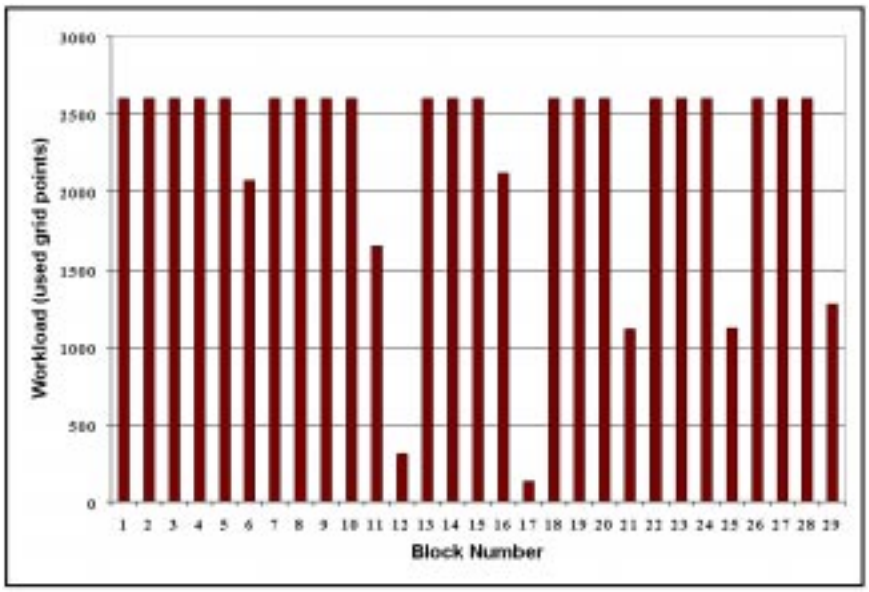

Figure 4: Workload for the 29DDG grid

choose the orientation of the block grid along the coastline so that the number of unused grid points in each block is minimized. The smallest number of used grid points in this technique is 1,263, and this increase results in an improved workload distribution (Figure 6) compared with the 29DDG grid. The number of unused grid points for this 29BLK grid is only 4,007 as compared with 11,015 of the 29DDG grid.

3 The U.S. West Coast Simulation

MGPOM is a three-dimensional (3-D), primitive equations, time dependent, σ coordinates, and free-surface coastal ocean circulation

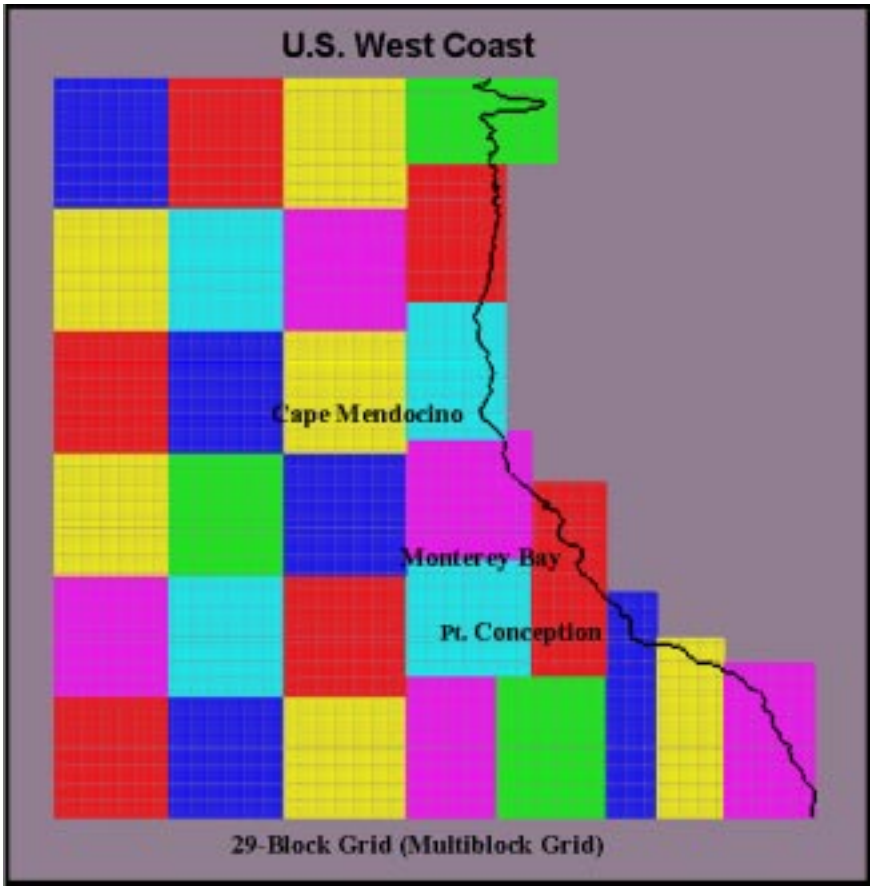

Figure 5: Coastline on the 29MBG grid

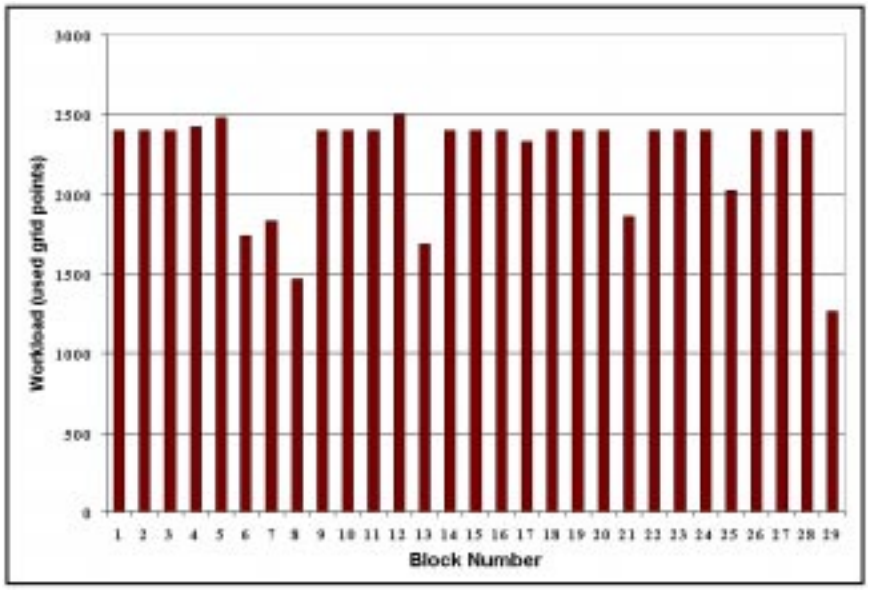

Figure 6: Workload for the 29MBG grid

model. The primitive equations in this model describe the velocity, surface elevation, salinity and temperature fields of the ocean. The ocean is assumed to be hydrostatic and incompressible. The model uses curvilinear orthogonal coordinates in the horizontal for the computational model grid. σ coordinates is used for the vertical direction from surface to the seafloor in which the coordinate is scaled on the water column depth. The model is embedded with a second moment turbulence closure submodel [5] to provide vertical mixing coefficients. The model has a free-surface and a split time-step. The external (barotropic) mode portion of the model is 2-D and uses the explicit numerical scheme for the time differencing. The

internal (baroclinic) mode is 3-D, and the vertical differencing is implicit. The latter condition eliminates time constraints for the vertical coordinate and permits the use of fine vertical grid resolution in the surface and the bottom boundary layers. More details of the model can also be found in [5].

Data for the U.S. west coast simulation were obtained from the Naval Oceanographic Office (NAVOCEANO) database. Bathymetry for the 01BLK, 29DDG, and 29MBG grids were computed by interpolation from the 2-min resolution bathymetry database. Initial temperature and salinity for these grids were also obtained by interpolation from the 10-min resolution temperature and salinity Generalized Digital Environmental Model (GDEM) database.

The U.S. west coast computational domain has three open boundaries. At these boundaries, the internal normal velocities are governed by a Sommerfeld radiation condition. The open-boundary condition for the surface elevation is zero gradient normal to the boundary. Temperature, salinity, and tangential velocities are upwinded at the open boundaries. The model is spun up for 30 days (diagnostic mode) in which the density distribution at all points on the computational grids is held fixed in time. The time-step scales used in this simulation are 40 sec for the external (barotropic) mode and 240 sec for the internal (baroclinic) mode. The time-step scale for the external mode is based on the CFL (Courant, Friedrichs and Lewy) condition and

the external wave speed, while the internal mode is based on the CFL condition and the internal wave speed.

After 30 days of diagnostic mode, the model is then run for 60 days. Numerical solutions after this 90-day simulation for the 01BLK grid, 29DDG grid, and the 29MBG grid yield identical results. Results of surface current computations between the 01BLK grid (Figure 7) and 29MBG grid (Figure 8) after the 90-day simulation were compared. These were found to be identical to each other. Numerical solutions for the surface temperature on the 01BLK grid (Figure 9) and the 29MBG grid (Figure 10) are also identical.

The serial version of MGPOM code for a 10-day simulation has an execution time of 67,000 sec on a single IBM SP processor. A parallel version of MGPOM uses MPI asynchronous sends and receives to exchange data between adjacent blocks at the interfaces. OpenMP [6] has been used as a second level of parallelization within each MPI process in a separate simulation of the Arabian Gulf [7]. In the current study, Pthreads is used as the second level of parallelism within each MPI process.

4 Pthreads Implementation

Pthreads is the library of POSIX standard functions for concurrent, multithreaded programming. The POSIX standard only defines an

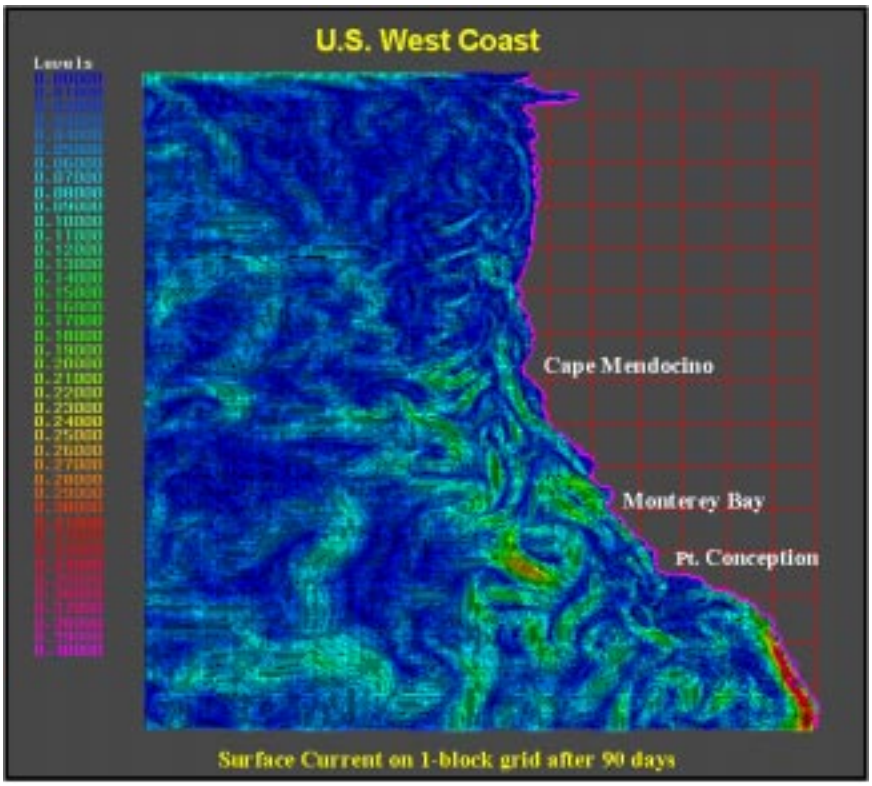

Figure 7: Surface current on the 01BLK grid

application programming interface (API) to the C programming language, not to Fortran. The FPTHRD package [8] consists of a Fortran module and file of C functions. The module defines Fortran derived types, parameters, interfaces, and routines to provide Fortran programmers the capabilities of the Pthread API. The C functions provide the interface from Fortran subroutine calls and map parameters into corresponding POSIX routines and function arguments.

The structure of the MGPOM code contains one major loop within

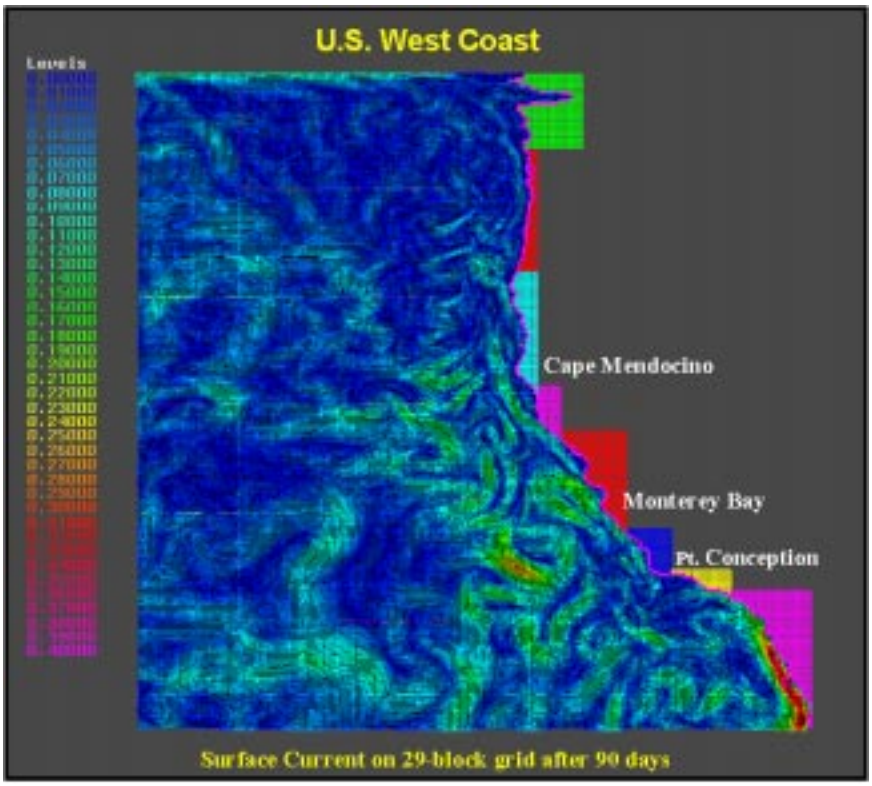

Figure 8: Surface current on the 29MBG grid

the main program. This loop iterates over the time-step of the execution simulation. Within this loop are several subroutine calls (four of these are for MPI communication, while the others perform computation). The computation subroutines are composed of multiple doubly and triply nested loops operating on 2- and 3-D arrays. Profiling the parallel MGPOM code revealed several routines that accounted for more than half of the wall-clock execution time in each processor. The top one among those routines is PROFQ, which takes nearly 20

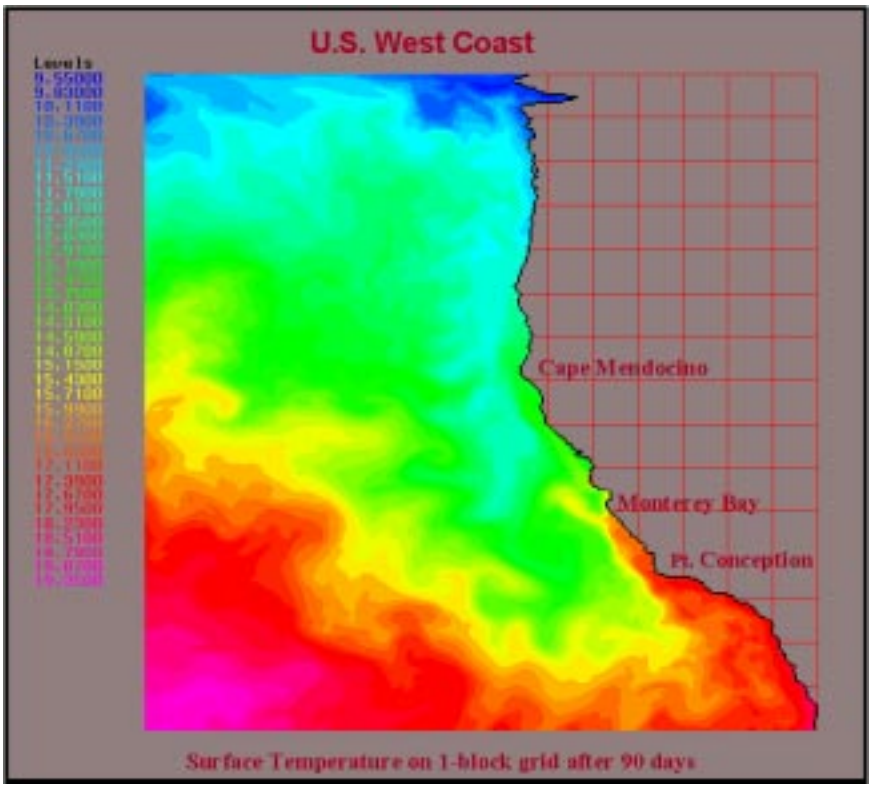

Figure 9: Surface temperature on the 01BLK grid

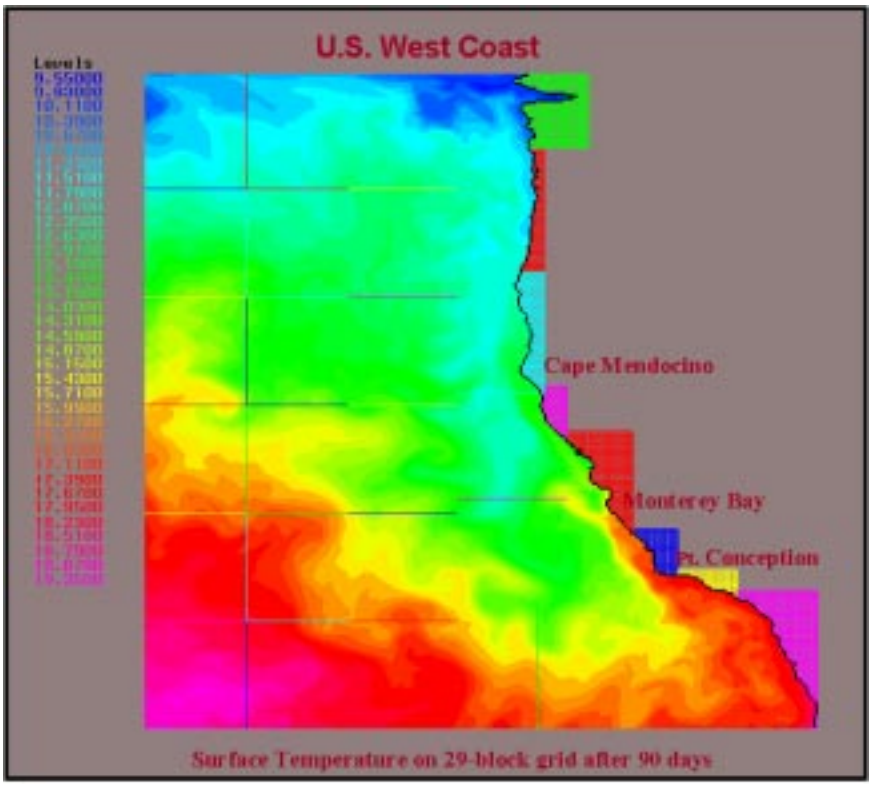

Figure 10: Surface temperature on the 29MBG grid

percent of the total execution time. This routine was the obvious first choice for threading.

Within each MPI process, based on the size of the block grid, a 2-D decomposition of the first two indices of the block into subblocks is then computed: one subblock is created per thread. The exact decomposition depends upon the number of threads assigned to the block and the relative lengths of the sides of the block. A decomposition that would yield the "squarest" subblocks is the ultimate goal. The index boundaries of the subblocks within the block grid are saved into a shared array and are used by each thread as loop iteration bounds within the PROFQ routine. After the block grid is decomposed into subblocks, the main routine is modified so that threads are created at run time according to the number of subblocks to be used within PROFQ.

5 Parallel Performance

While MGPOM processes use asynchronous communication, the processes must synchronize to some degree; e.g., processes with a small number of used grid points within assigned blocks are forced to wait on the actual receipt of data from processes with large blocks that have been performing more calculations prior to communication. The greater the difference in size between adjacent blocks, the larger the

load imbalance of computation will be. In order to quantify the degree of load imbalance within a given code segment, *idle overhead* is defined as the ratio of total execution time to maximum possible execution time expressed as a percentage:

$$\text{idle overhead} = 100\% \times \left(1. - \frac{\sum_{i=1}^{N} t_i}{N \times t_{max}}\right) \tag{1}$$

where N is the number of MPI processes, t_i is execution time of process i, and t_{max} is the largest time t_i.

All results presented herein in terms of idle overhead and cumulative time are for the PROFQ routine on the 29DDG and 29MBG grids.

The cumulative timing results of PROFQ for the MPI-Only code version for a run of 10 simulated days on the 29DDG grid (Figure 11) show an idle overhead of 13 percent. The total wall-clock execution time of this run was 2,622 sec, a 26 times speedup, as compared to the 01BLK grid serial time, when run on 29 IBM SP processors. Idle overhead is 8 percent for the 29MBG grid (Figure 12) within the PROFQ routine. The total wall-clock execution time of this run was 2,018 sec and a 33 times speedup was achieved for this grid.

The cumulative timing results of PROFQ for the MPI-Pthreads code version (Figure 13) yields 7 percent of idle overhead time for the 29DDG grid. This run was performed on 29 IMB SP SMP nodes

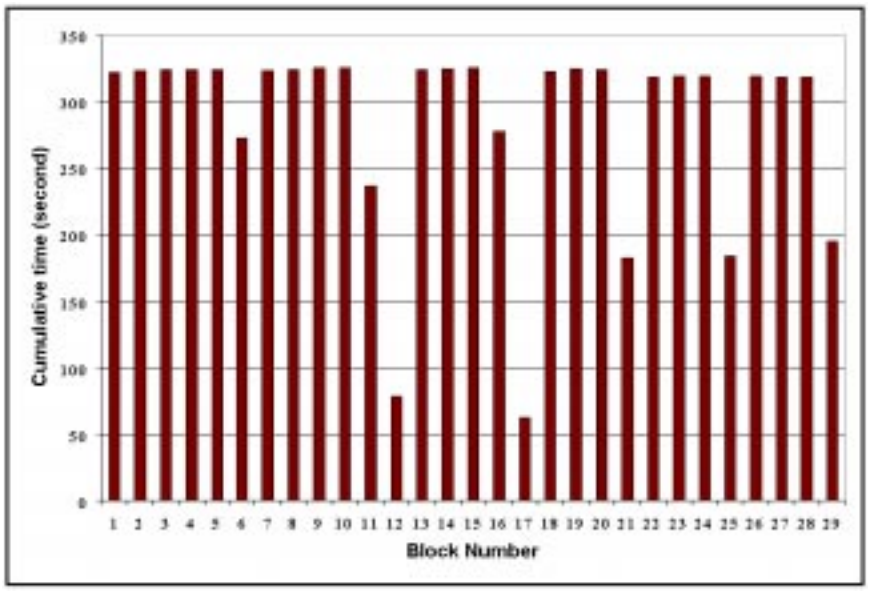

Figure 11: PROFQ:(MPI-Only) cumulative execution time in seconds

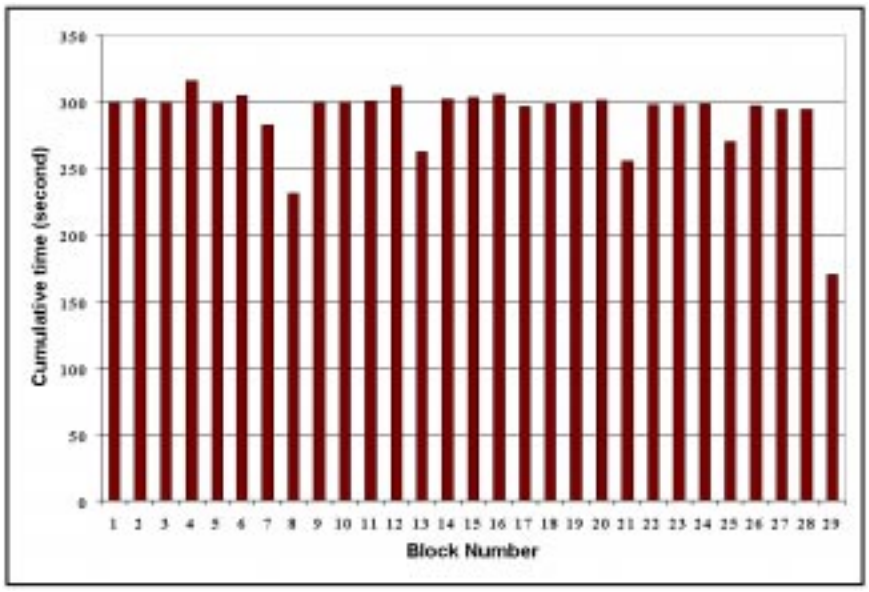

Figure 12: PROFQ:(MPI-Only) cumulative execution time in seconds

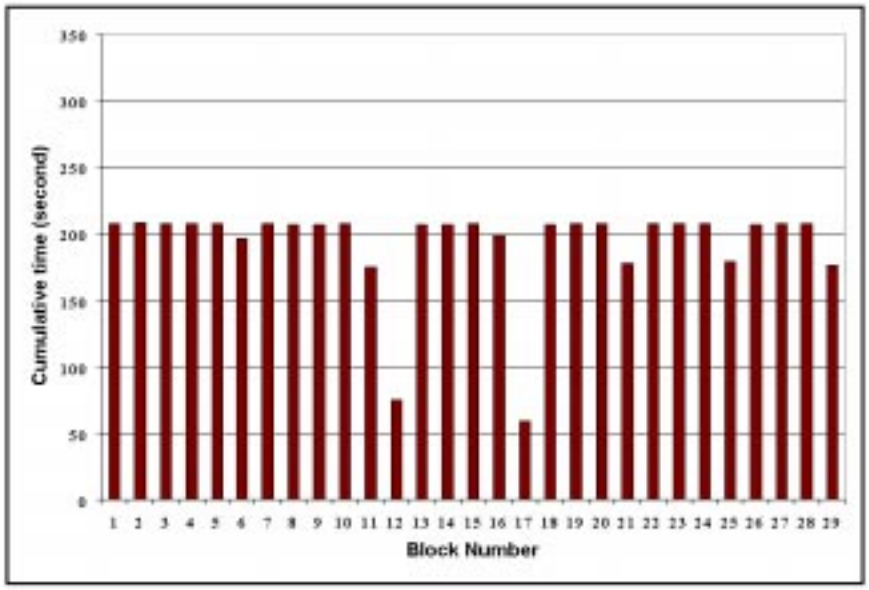

Figure 13: PROFQ:(MPI-Pthreads) cumulative execution time in seconds

(each with four processors) using a maximum of two threads per process (58 processors) based on assigned workload. The total wall-clock execution time of this run was 1,720 sec, a 39 times speedup over the one-block version. Only 4 percent of idle overhead time was committed within the PROFQ routine (Figure 14) for the MPI-Pthreads code version on the 29MBG grid. The total wall-clock execution time of this run was 1,625 (also run on 29 IBM SP four-processor nodes), a 41 times speedup over the serial version.

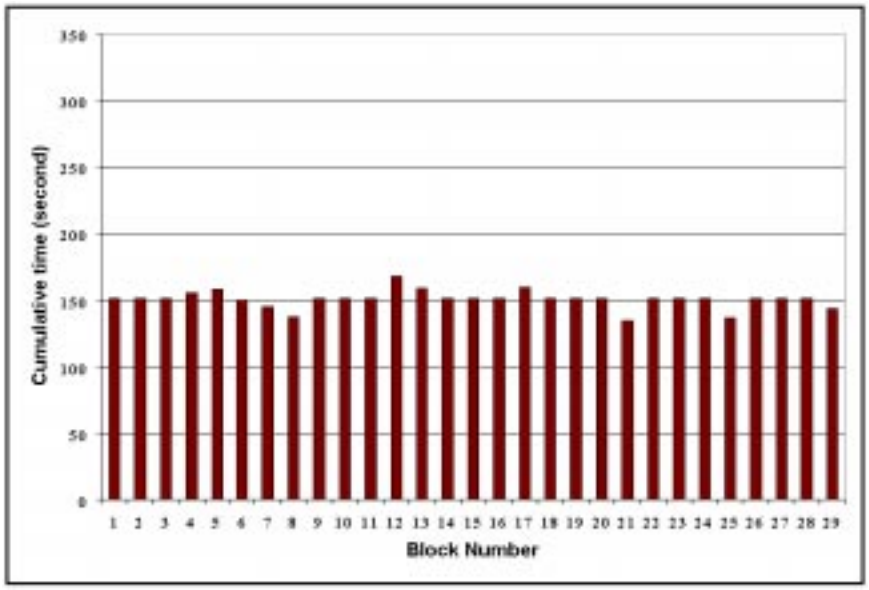

Figure 14: PROFQ:(MPI-Pthreads) cumulative execution time in seconds

6 Conclusions

The timing results show a significant improvement in the execution time as well as in the load imbalance produced by MPI-Only execution. Through use of the hand-coded, dynamic threading within Pthreads, load imbalance between the MPI processes of the two grids can be improved. The performance results achieved were the results of threading a single MGPOM routine. Other candidate routines within the code are still available.

Acknowledgment

This work was funded by the DoD High Performance Computing Modernization Program U.S. Army Engineer Research and Development Center (ERDC) Major Shared Resource Center through Programming Environment and Training (PET), supported by Contract Number: DAHC 94-96-C0002, Computer Sciences Corporation.

References

[1] Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J., *MPI—The Complete Reference: Volume 1, the MPI Core*, MIT Press, Cambridge, 1998.

[2] Oberpriller, W. D., Sawdey, A. C., O'Keefe, M. T., and Gao, S., "Paralleling the Princeton Ocean Model using TOPAZ," `http://topaz.lcse.umn.edu`.

[3] Luong, P. V., Breshears, C. P., and Ly, L. N., "Application of Multiblock Grid and Dual-Level Parallelism in Coastal Ocean Circulation Modeling," *Journal of Applied Mathematical Modeling*, In Review.

[4] Stokes, M., Jiang, M., and Remotique, M., "EAGLEview Grid Geration Package," EAGLEView Version 2.4 Manual. Missisippi State University/National Science Foundation Engineering Research Center for Computational Field Simulation, December 1992.

[5] Blumberg, A. F., and Mellor, G. L., "A Description of a Three-Dimensional Coastal Ocean Circulation Model." In *Three-Dimensional Coastal Models,* Coastal and Estuaries Sciences. Heaps, N. S., editor, AGU Geophysical Monograph Board, 1987, 1.

[6] OpenMP Architecture Review Board, "OpenMP Fortran Application Program Interface, Version 1.0," `http://www.openmp.org`, October 1997.

[7] Luong, P. V., Breshears, C. P., and Ly, L. N., "Dual-Level Parallelism and Multiblock Grids in Coastal Ocean Circulation Modeling," Technical Report ERDC MSRC/PET, TR/00-08, Feb. 2000.

[8] Hanson, R. J., Breshears, C. P., and Gabb, H. A., "A Fortran Interface to POSIX Threads," Technical Report ERDC MSRC/PET, TR/00-18, Feb. 2000.